11-02-00

JC957 U.S. PTO 10/31/00

PTO/SB/05 (08-00)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this box → [X]

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.	1411
First Inventor	DUJARI
Title	Dynamic Network Cache Directories
Express Mail Label No.	EE591449807US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

1. [X] Fee Transmittal Form (e.g., PTO/SB/17) *(Submit an original and a duplicate for fee processing)*
2. [ ] Applicant claims small entity status. See 37 CFR 1.27.
3. [X] Specification [Total Pages 29] + cover sheet *(preferred arrangement set forth below)*
   - Descriptive title of the invention
   - Cross Reference to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to sequence listing, a table, or a computer program listing appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings *(if filed)*
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
4. [X] Drawing(s) *(35 U.S.C. 113)* [Total Sheets 6]
5. Oath or Declaration [Total Pages 2]
   a. [ ] Newly executed (original or copy)
   b. [X] Copy from a prior application (37 CFR 1.63 (d)) *(for continuation/divisional with Box 17 completed)*
      i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
6. [ ] Application Data Sheet. See 37 CFR 1.76
7. [ ] CD-ROM or CD-R in duplicate, large table or Computer Program *(Appendix)*
8. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. [ ] Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. [ ] CD-ROM or CD-R (2 copies); or
      ii. [ ] paper
   c. [ ] Statements verifying identity of above copies

jc917 U.S. PTO 09/703381 10/31/00

ACCOMPANYING APPLICATION PARTS

9. [ ] Assignment Papers (cover sheet & document(s))
10. [ ] 37 CFR 3.73(b) Statement *(when there is an assignee)* [ ] Power of Attorney
11. [ ] English Translation Document *(if applicable)*
12. [ ] Information Disclosure Statement (IDS)/PTO-1449 [ ] Copies of IDS Citations
13. [ ] Preliminary Amendment
14. [X] Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
15. [ ] Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
16. [X] Other: Exp. Mail Cert. and Transmittal, Petition for 1 mo. Ext., Credit Card Payment form in parent case

17. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:*

[ ] Continuation [X] Divisional [ ] Continuation-in-part (CIP) of prior application No.: 09 / 058,982

Prior application information: Examiner MOAZZAMI, N. Group / Art Unit: 2759

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS

[ ] Customer Number or Bar Code Label (Insert Customer No. or Attach bar code label here) or [ ] Correspondence address below

Name	MICHALIK & WYLIE, PLLC				
Address	14645 Bel-Red Road Suite 103				
City	Bellevue	State	WA	Zip Code	98007
Country	USA	Telephone	425-653-3520	Fax	206-653-3603

Name (Print/Type)	Albert S. Michalik	Registration No. (Attorney/Agent)	37,395
Signature	Albert S. Michalik	Date	10/31/00

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231 DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

Please type a plus sign (+) inside this box → [+]

PTO/SB/21 (6-98)
Approved for use through 09/30/2000. OMB 0651-0031
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/058,982
Filing Date	April 10, 1998
First Named Inventor	DUJARI
Group Art Unit	2759
Examiner Name	MOAZZAMI
Attorney Docket Number	1410

Total Number of Pages in This Submission: 3

09/703381 U.S. PTO 10/31/00

ENCLOSURES *(check all that apply)*

- ☐ Fee Transmittal Form
- ☒ Fee Attached
- ☐ Amendment / Response
- ☐ After Final
- ☐ Affidavits/declaration(s)
- ☒ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
- ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Assignment Papers *(for an Application)*
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition Routing Slip (PTO/SB/69) and Accompanying Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Small Entity Statement
- ☐ Request for Refund
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Additional Enclosure(s) *(please identify below)*:

Credit Card Payment form in the amount of $110.00

Remarks: **Please note that this is being filed concurrently with a divisional patent application claiming priority to the subject application.**

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Albert S. Michalik, Reg. No. 37,395
Signature	Albert S. Michalik
Date	October 31, 2000

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on this date: 10/31/2000

Typed or printed name	Albert S. Michalik		
Signature	Albert S. Michalik	Date	October 31, 2000

PTO/SB/17 (08-00)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2000

Patent fees are subject to annual revision.

TOTAL AMOUNT OF PAYMENT ($) 998.00

Complete if Known	
Application Number	unassigned
Filing Date	10/31/2000 via exp. mail #EE591449807US
First Named Inventor	DUJARI
Examiner Name	unassigned
Group Art Unit	unassigned
Attorney Docket No.	1411

METHOD OF PAYMENT (check one)

1. ☐ The Commissioner is hereby authorized to charge indicated fees and credit any overpayments to:

Deposit Account Number

Deposit Account Name

☐ Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

☐ Applicant claims small entity status. See 37 CFR 1.27

2. ☒ **Payment Enclosed:**

☐ Check ☒ Credit card ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	690	201	345	Utility filing fee	710
106	310	206	155	Design filing fee	
107	480	207	240	Plant filing fee	
108	690	208	345	Reissue filing fee	
114	150	214	75	Provisional filing fee	

SUBTOTAL (1) ($) 710.00

2. EXTRA CLAIM FEES

		Extra Claims		Fee from below	Fee Paid
Total Claims	36	-20** =	16	x 18	= 288
Independent Claims	3	- 3** =	0	x 80	= 0
Multiple Dependent				0	= 0

**or number previously paid, if greater; For Reissues, see below

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	78	202	39	Independent claims in excess of 3
104	260	204	130	Multiple dependent claim, if not paid
109	78	209	39	** Reissue independent claims over original patent
110	18	210	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 288

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for *ex parte* reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	
116	380	216	190	Extension for reply within second month	
117	870	217	435	Extension for reply within third month	
118	1,360	218	680	Extension for reply within fourth month	
128	1,850	228	925	Extension for reply within fifth month	
119	300	219	150	Notice of Appeal	
120	300	220	150	Filing a brief in support of an appeal	
121	260	221	130	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive - unavoidable	
141	1,210	241	605	Petition to revive - unintentional	
142	1,210	242	605	Utility issue fee (or reissue)	
143	430	243	215	Design issue fee	
144	580	244	290	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	690	246	345	Filing a submission after final rejection (37 CFR § 1.129(a))	
149	690	249	345	For each additional invention to be examined (37 CFR § 1.129(b))	
179	690	279	345	Request for Continued Examination (RCE)	
169	900	169	900	Request for expedited examination of a design application	

Other fee (specify) _____

* Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 0

SUBMITTED BY

Complete (if applicable)

Name (Print/Type)	Albert S. Michalik	Registration No. (Attorney/Agent)	37,395	Telephone	4256533520
Signature	Albert S. Michalik			Date	10/31/00

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231

PATENT
Attorney Docket No. 1411

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Divisional Application (of prior application 09/058,982; GAU 2759; Examiner MOAZZAMI)

Applicant: DUJARI, R.

Title: METHOD AND SYSTEM FOR BALANCING DIRECTORIES (as amended herein)

Expr. Mail: EE591449807US

Date: October 31, 2000

PRELIMINARY AMENDMENT

Assistant Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

Prior to the examination of the above-identified patent application, please enter the following amendment.

IN THE SPECIFICATION:

Please amend the title by deleting "DYNAMIC NETWORK CACHE DIRECTORIES" on the cover sheet, and on page 1, line 1, and substituting therefor -- METHOD AND SYSTEM FOR DIRECTORY BALANCING --.

Please amend the specification by inserting before the "FIELD OF THE INVENTION" section a new section entitled:

-- CROSS-REFERENCE TO RELATED APPLICATION

This application is a divisional of copending United States Patent Application Serial No. 09/058,982, filed April 10, 1998. --

REMARKS

Entry of the present preliminary amendment is respectfully requested. If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney.

Respectfully submitted,

Albert S. Michalik

Albert S. Michalik, Registration No. 37,395
Attorney for Applicants
Michalik & Wylie, PLLC
14645 Bel-Red Road
Suite 103
Bellevue, Washington 98007
(425) 653-3520 (telephone)
(425) 653-3603 (facsimile)

Date: October 31, 2000

1411 Prelim amendment

S P E C I F I C A T I O N

TO ALL WHOM IT MAY CONCERN:

Be it known that I, Rajeev Dujari, a citizen of the United States, residing at 65 Kirkland Avenue #305, Kirkland, Washington 98033, have invented a certain new and useful **DYNAMIC NETWORK CACHE DIRECTORIES** of which the following is a specification.

DYNAMIC NETWORK CACHE DIRECTORIES

FIELD OF THE INVENTION

The invention relates generally to computer systems and networks, such as intranets and the Internet, and more particularly to reducing security risks of cached network content without compromising usability.

BACKGROUND OF THE INVENTION

For network client applications, such as web browsers, a limiting performance factor is often low bandwidth to the server. To mitigate this low-bandwidth problem, network client applications often cache content replicated from servers, so that as much information as possible is kept available on the client user's hard drive. To cache content, the local machine generates a filename from the content's URL (Uniform Resource Locator) and stores the file in a cache directory (folder). As data access times from the hard drive are typically orders of magnitude faster than download times, some or all of a server's content may often be rapidly accessed from the cache with little or no downloading of data from the server. In the extreme case, the computer or server may be offline from the network, in which case the cache may still provide some version of the content. Note that caching

operations are automatic and invisible to the user, and thus no security checks (e.g., code signing verification) are immediately performed on the downloaded content. However, content that is cached is harmless unless opened.

While content caching thus provides substantial performance improvements, a big security problem is that a malicious web site may easily guess the default location of the cache and the filename generated for a given URL. By including a page with an embedded http: reference to a virus or other malicious program, the malicious site causes the virus / malicious program to be automatically downloaded to the cache. The site and/or page may also embed a guessed file: reference to the cache location of the virus. Note that normal security checks are carried out if the user invokes the http: reference, since the operating system recognizes the content as coming from a server. However, if the user invokes the guessed file: reference, (e.g., by clicking a corresponding location on the page or in some other manner), the operating system treats this as any other local file in the file system, thus executing or opening the virus / malicious program. As can be readily appreciated, normal code signing verification techniques applied to downloaded programs may be bypassed in this manner.

By way of example, assume via an embedded http: reference

such as http://server/virus.exe, a malicious site places a hypothetical file named "virus.exe" in a user's cache directory named (e.g., by default) "C:\Windows\Temporary Internet Files\Cache2". If the site correctly guesses this file and location, the malicious site may include a file: reference, i.e., "file://c:\windows\Temporary Internet Files\Cache2\virus.exe" on the same (or even another) page. When the user invokes this file: reference, the virus program is executed.

Some contemporary web browsers solve this security problem by generating random filenames for cached files, whereby to be able to invoke the file via a corresponding file: reference, the site would have to guess the filename from an extremely large number of permutations. However, this has the drawback that applications (e.g., Microsoft Word) which are invoked from valid downloaded content will display and may even remember the random file names, confusing users.

SUMMARY OF THE INVENTION

Briefly, the present invention provides a system and method of storing content in a cache in a manner that makes it virtually impossible for a site to guess the cache location. To this end, random subdirectory names are generated for one or more caches, and randomly-named cache directories are

created from the random subdirectory names. When content is downloaded from a server, the content is stored as one or more files in one of the randomly-named cache directories. In addition to generating random subdirectory names, the system and method provide for enhanced file system performance by balancing the number of files among the cache directories, and by limiting the number of files in any cache directory by creating additional cache directories as needed.

Other advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram representing a computer system into which the present invention may be incorporated;

FIG. 2 is a block diagram representing a general conceptual model of the present invention;

FIG. 3 is a block diagram generally representing various components for implementing the method and system of the present invention;

FIG. 4 is a representation of a table maintained for relating site references to locations and names of locally cached files;

FIG. 5 is a flow diagram generally representing the steps taken for generating random subdirectory names and creating cache directories therefrom;

FIG. 6 is a representation of an indexed table maintained for locating cache subdirectory names and a count of the number of files in each corresponding cache directory; and

FIG. 7 is a flow diagram generally representing the logic for balancing files among cache subdirectories.

DETAILED DESCRIPTION

Exemplary Operating Environment

Figure 1 and the following discussion are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor-based or programmable consumer

electronics, network PCs, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20 or the like, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including the system memory to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 may further include a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading

from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD-ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read-only memories (ROMs) and the like may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35, (including a file system therein and/or associated therewith), one or more application programs 36, other program modules 37 and program data 38. A

user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor 47, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 20, although only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking

environments are commonplace in offices, enterprise-wide computer networks, Intranets and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the local network 51 through a network interface or adapter 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the wide area network 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Dynamic Network Cache Directories

FIG. 2 shows a generalized conceptual model of the present invention wherein a network application 60 such as a browser in a client machine (e.g., the personal computer system 20) communicates with a server (e.g., the remote computer 49) in order to download content 62 therefrom. Communication between the client 20 and the server 49 may take

place using one of several well-known network protocols, such as hypertext transfer protocol (HTTP), file transfer protocol (FTP), Common Internet File System (CIFS) protocol, or Gopher, although for purposes of simplicity, the invention will be primarily described with respect to HTTP. Content available through these protocols may also be downloaded from the server to the client by alternative means, such as a multicast protocols or CD-ROM installation, for example. As used herein, "server" or "network server" includes any machine or combination of machines having content thereon. Network servers may thus include http "web sites," including those having sites with different names (which may be regarded as different virtual servers even if they are hosted on the same physical machine). Note that a web site may be distributed over many virtual servers, which in turn may be distributed over many physical machines.

In any event, the network application 60 includes or otherwise accesses a cache manager component 64 that includes code for caching some or all of the content 62, ordinarily via application programming interface (API) calls through APIs 65 to the operating / file system 35. Each distinctly-referenced portion of a server's content 62 is stored as a file in one or more cache directories 66_1 - 66_n. Note that some or all of the various components referenced herein may be combined with or

included within other components, while others may be separate from one another and be appropriately invoked as needed. For example, the cache manager component 64 may be part of the network application 60 (e.g., browser) code, or may be a separate component, (e.g., object, dynamic link library function and so on) that other network applications may call on.

To store and retrieve cached files, as shown in FIG. 3 the cache manager component 64 includes a mechanism 68 for converting server references (URLs) to local file system filenames. Although URL names and the like provided by servers often resemble filenames, certain characters in the URL name may not be allowed in a particular file system, and thus the converter 68 substitutes appropriate characters as necessary. Also, the name may be decorated, say by a appending a number, to distinguish it from a file for a similar but different URL. Note that the converter could generate random file names, however from the user's viewpoint, the use of user-friendly file names provides many advantages over randomized filenames, and thus the preferred converter attempts to match names as closely as possible to their URL names. In any event, after converting the names to corresponding file names, the server references and their corresponding local filenames are maintained in a table 70

(FIG. 4) or the like. As described in more detail below, the table 70 facilitates the use of multiple cache directories.

As shown in FIG. 3, the cache manager 64 further includes a cache handling mechanism 72 to facilitate caching operations, including creating and removing files and directories, and opening, reading and writing files. To this end, the cache handling mechanism 72 uses the filename table 70 and/or the converter mechanism 68 as needed to determine appropriate file names for interfacing with file system APIs.

In accordance with one aspect of the present invention and as generally represented in FIG. 3, the cache manager component 64 generates random subdirectory names for the caches 66_1 - 66_n. To this end, the cache manager 64 includes (or accesses) code referred to herein as a subdirectory name generator component 74. At a time when no cache exists or when one or more additional caches are needed (as described below), the subdirectory name generator component 74 along with a cache balancing mechanism 76 (as also described below) determines how many (possibly additional) caches to create, creates that many caches (cache directories), and indexes the caches in a cache index table 78, also described below. In this manner, any number of caches may be created. To generate the random names of the cache subdirectories, the subdirectory name generator component 74 obtains random numbers from a

random number generator 80, converts each random number to an alphanumeric character, and assembles a name from those alphanumeric characters. The subdirectory name generator component 74 also stores the subdirectory names in the indexed cache table 78 so that the cache directories may be efficiently represented by an index number, (e.g., a single byte or word, depending on how many caches are allowed), and thus quickly located. Furthermore, the indexing table may allow the cache directory names to be changed efficiently even after the directories have been created.

More particularly, as generally represented in the flow diagram of FIG. 5 beginning at step 500, the subdirectory name generator component 74 accesses the cache table 78 to determine how many indexed cache subdirectories already exist, (if any), and sets a starting cache count based on this number. For example, the first time one or more caches are created, none exist, whereby the cache count is set to zero, while if caches zero to three are already indexed, the cache count is set to four. Also at step 500, the subdirectory name generator component 74 determines the last cache (index) number to create based on the additional number of cache subdirectories desired. At present, for purposes of efficiency, cache subdirectories are created four at a time, whereby if no caches exist, the last cache number is set to

three, while if four caches (zero through three) already exist, the last cache number will be seven at step 500. Moreover, although not shown in FIG. 5, if the number of cache subdirectories exceed some predetermined value, (e.g., sixty-four), the subdirectory name generator component 74 may decide not to create any additional caches.

At step 502, a pointer is set to point to the first character in a string that will ultimately become the name of the subdirectory. The string is preferably eight characters in length, (for efficiency purposes with certain file systems), but alternatively may be eight characters plus a three-character extension, or some other number of characters. An even larger number of permutations could be generated by using longer directory names or using nested subdirectories, (and in particular, although not necessarily, nested subdirectories that are also randomly named), for example "FDSWOSI\SFW2HN54". Moreover, note that the cache directory is still randomly named as long as the cache directory name is based on some generated random subdirectory name or part thereof, since such a modified random subdirectory name is also a random subdirectory name. For example, if the string "H93UGUB1" is randomly generated, then as long as the cache directory name includes in its path some or all of the name (e.g., \H93UGUB1-1, \H93UGUB1-2, or \3UGU), the cache

directory name is still random. Indeed, although not necessary to the present invention, the length of the string may itself be randomly generated, such as to be anywhere between five and eight characters. For purposes of the present example, eight character subdirectory names, with no extension, will be used.

In keeping with the present invention, at step 504, a random number is obtained from the random number generator 80. At present, the random number that is generated (including any necessary conversion, such as via multiplication and rounding) is between zero and thirty-five. At step 504, the number is converted to an alphanumeric character and the ASCII character is written into the string. Preferably, a number from zero through nine is converted to an ASCII character value "0" through "9", respectively while a number of ten through thirty-five is converted to an ASCII character value of "A" through "Z", respectively. Thus, if the random number was thirty-three, the character is "X" in ASCII. As can be appreciated, this provides 36^8 possible character permutations. Step 508 tests if the string is full, i.e., in the present example, if the string contains all eight characters of an eight character name. If not full, the string pointer moves forward at step 510 and the character generation process of steps 504 - 506 are repeated.

Eventually, the string will be full and step 508 will branch to step 512 wherein the cache directory will be created from the randomized string using an appropriate file system API or the like. Also at step 512, the random subdirectory name (the string "XDF53GBI") is indexed based on the value of the cache counter, i.e., at location zero, as represented in the exemplary cache table 78 of FIG. 6. For example, the first cache subdirectory created (indexed by zero) may be named "XDF53GBI", whereby the cache directory may become, for example, "C:\Windows\Temporary Internet Files\XDF53GBI\". Note that the full path need not be indexed as long as the folders that make up the preceding portion of the path are the same for all of the caches 66_1 - 66_n. As can be readily appreciated, such a full pathname is highly unlikely to be guessed by a remote site, thus solving the security problem that arises when the cache location and filename is known, as described above. Note that if an error occurs in creating the subdirectory, the error may be handled at this time. For example, if by (the very unlikely) chance the named subdirectory already exists, the indexing sub-step may be skipped and the cache counter may be decremented (not shown), whereby another random subdirectory would be created in its place. However, since at least one cache subdirectory would have to already exist to generate such an error, after the

indexing sub-step is skipped, the error otherwise may be essentially ignored, since it does not (practically) matter how many unique cache subdirectories are actually created at a given time.

Step 514 then determines whether all desired cache subdirectories have been created. If so, the cache subdirectories have been created and indexed, and the name generation process ends. If not, the cache count is incremented at step 516 and the process repeated for a new string by returning to step 502.

Although not necessary to the invention, for purposes of efficiency the cache manager 64 may further include a cache balancing mechanism 76 to distribute the content to be cached among the caches 70_1 - 70_n and/or create additional caches as necessary. Note that the performance of certain file systems begins to degrade when more than a certain number of files (e.g., one-thousand) are in the same subdirectory, and thus the balancing mechanism 76 operates to avoid such a situation. To this end, the balancing mechanism 76 tracks the number of files in each directory and determines in which cache a server's downloaded content should be cached. At the same time, the balancing mechanism 76 may use the count to determine whether the existing caches have enough files, and thus whether more caches need to be created. As shown in FIG.

6, the count may be maintained as a field in the cache table 78. Note that the balancing mechanism 76 described herein caches a server's content in one or more files in the same cache subdirectory. As can be appreciated, however, the balancing mechanism 76 may, for example, alternatively distribute the files from a given server's content among the various caches.

FIG. 7 shows exemplary logic for the balancing mechanism 78, which operates when a server's files are to be cached. Beginning at step 700, the balancing mechanism 76 walks through the index and determines which cache has the lowest number of files therein. In the present example shown in FIG. 6, the cache with an index of one corresponding to the subdirectory named "H3A38LPR" has the least number (329) of files, and thus this cache is selected. If the number of files in this cache plus the number to be cached is below some predetermined threshold amount of files (e.g., one-thousand) then step 702 branches to step 704 where the server's files are downloaded to the cache (e.g., "C:\Windows\Temporary Internet Files\H3A38LPR") and the count of files in the cache table 78 increased to reflect the new number of files therein. As can be appreciated, in this manner, the mechanism 76 distributes the files among the various caches based on the subdirectory that has the least number of files thereunder.

Step 708 stores the cache:filename in the URL to cache:filename table 70 so that the location of the cached file may be quickly determined from its server reference. Note that to efficiently store the name, the cache subdirectory is identified by its index rather than by its eight-character subdirectory name or full cache directory path name.

If instead at step 702 the number of files in the selected cache plus the number to be cached exceeds the predetermined threshold amount, step 702 branches to step 706 to determine if more caches (subdirectories) may be created. Step 706 compares the number of cache subdirectories against some predetermined maximum number (e.g., sixty-four), and if the number is below this value, the balancing process 76 branches to step 708. Step 708 creates the new subdirectories (e.g., by invoking the subdirectory name generator 72 which executes the steps of FIG. 5), and returns to step 700 where one of the new, empty caches are selected. If no more subdirectories may be created at step 706, then step 710 is executed, for example, to remove files from existing caches (such as those which have not been accessed for the longest time). Note that a similar-such removal of files from a cache is a well-known operation when the total space allocated for caching files becomes filled up. However, unlike simple

space-based removal, in addition to removing the files, step 710 also appropriately reduces the file count maintained in the cache index 78. Note that in step 710, the cache and number of files to remove therefrom may be based on an algorithm, e.g., remove a number m of the least-recently accessed files from the cache having the largest amount of files therein, where m is the number of new files to be cached.

Lastly, it should be noted that the balancing operation is concerned with the number of files in each cache subdirectory, and not the total size of the files in each subdirectory. Moreover, whenever another process modifies the number of files in a cache directory, the corresponding file count in the index 78 need to be appropriately adjusted.

While the invention is susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

WHAT IS CLAIMED IS:

1. A computer-implemented method, comprising:

providing at least two selected directories for storing files; and

automatically balancing files among each of the selected directories.

2. The computer-implemented method of claim 1 further comprising, receiving information corresponding to a new file to store.

3. The computer-implemented method of claim 1, wherein providing at least two selected directories for storing files includes automatically creating at least one directory.

4. The computer-implemented method of claim 1, wherein automatically balancing files among each of the selected directories includes determining which of the directories has a least number of files therein.

5. The computer-implemented method of claim 1, wherein automatically balancing files among each of the selected directories includes determining when a selected directory has a number of files stored therein that exceeds a limit.

6. The computer-implemented method of claim 5, further comprising, receiving information corresponding to a new file to store, determining that each selected directory has a number of files therein that exceeds a limit, and automatically creating at least one new selected directory.

7. The computer-implemented method of claim 1 further comprising, for each file, tracking which selected directory that file is stored in.

8. The computer-implemented method of claim 1 further comprising, maintaining a count of a number of files stored in each selected directory.

9. The computer-implemented method of claim 1 wherein at least one of the selected directories caches content downloaded from a server.

10. The computer-implemented method of claim 1 wherein at least one of the selected directories is randomly named.

11. The computer-implemented method of claim 1 wherein each of the selected directories is randomly named, and wherein each of the selected directories caches content downloaded from a server.

12. The computer-implemented method of claim 11 further comprising, maintaining a table including server content references and filenames converted therefrom.

13. The computer-implemented method of claim 1, wherein automatically balancing files among each of the selected directories includes determining a selected directory having a lowest file count, and moving files from another selected directory to the selected directory having the lowest file count.

14. The computer-implemented method of claim 1, further comprising, maintaining an index including a directory name for each selected directory, and for each directory name, maintaining a file count of a number of files stored therein.

15. The computer-implemented method of claim 1, further comprising, comparing the number of files in a selected directory having the least number of files therein against a predetermined threshold value, and based on the comparison, generating at least one additional selected directory.

16. The computer-implemented method of claim 1, further comprising, maintaining an indexed directory table including data corresponding to the selected directories therein, and maintaining a table including file information and corresponding file directory information for each file in one of the selected directories.

17. The computer-implemented method of claim 1, wherein automatically balancing files among each of the selected directories includes moving at least one file from one selected directory to another directory following deletion of at least one other file.

18. The computer-implemented method of claim 1, further comprising maintaining a file count of a number of files stored in each selected directory, and wherein automatically balancing files among each of the selected directories includes moving at least one file out of a selected directory

when the file count maintained therefor is below a threshold value.

19. The computer-implemented method of claim 18, further comprising removing a selected directory based on the file count maintained therefor.

20. In a computing device having a file system, a system, comprising:

a balancing mechanism configured to automatically create a selected directory in the file system for storing files, and further configured to distribute files from at least one other directory in the file system to the selected directory; and

a data structure, the data structure maintained by the balancing mechanism to track information on the files in the selected directory.

21. The system of claim 20 wherein the at least one other directory comprises a directory created by the balancing mechanism.

22. The system of claim 20 wherein the balancing mechanism receives information corresponding to a new file to store.

23. The system of claim 20, wherein the balancing mechanism distributes the files based on a number of files in the at least one other directory.

24. The system of claim 20, wherein the balancing mechanism creates an additional directory for distributing files thereto.

25. The system of claim 24, wherein the balancing mechanism receives information corresponding to a files to store, and distributes those files based on a file count of the selected directory and the additional directory.

26. The system of claim 24 further comprising, a table having information therein indicating which directory each file is stored in.

27. The system of claim 24, wherein the balancing mechanism is configured to create the additional directory upon a determination that the selected directory has a number of files stored therein that exceeds a limit.

28. The system of claim 20 wherein the data structure includes a count of a number of files stored in each selected directory.

29. The system of claim 20 wherein the selected directory caches content downloaded from a server.

30. The system of claim 20 wherein the selected directory is randomly named.

31. The system of claim 20 wherein the selected directory is randomly named, and wherein the selected directory caches content downloaded from a server.

32. The system of claim 20 wherein the data structure includes a directory name for the selected directory and a file count of a number of files stored in the selected directory.

33. The system of claim 20 wherein the balancing mechanism is further configured to move at least one file from one selected directory to another selected directory in response to deletion of at least one other file.

34. The system of claim 20 wherein the data structure tracks a file count of a number of files stored in each selected directory, and wherein the balancing mechanism is further configured to move at least one file out of a selected directory when the file count maintained therefor is below a threshold value.

35. The system of claim 34, wherein the balancing mechanism removes a selected directory based on the file count maintained therefor.

36. A computer-readable medium having stored thereon a data structure, comprising:

a first field identifying one of a plurality of directories; and

a second data field including data corresponding to a number of files stored in the directory identified in the first data field, the second data field updated as files are moved among the plurality of directories.

ABSTRACT

A system and method for virtually eliminating a potential security risk inherent with the caching of network content. Random subdirectory names are generated for cache directories, while continuing to generate user-friendly file names. Security is achieved since malicious sites cannot guess the cache location, which if guessed along with the filename, could cause a user connected to the site to inadvertently execute malicious content downloaded by the site to the cache. In addition to generating random subdirectory names, the system and method provide enhanced performance by balancing the number of files among the cache directories, and by limiting the number of files in any cache directory by creating additional cache directories as needed.

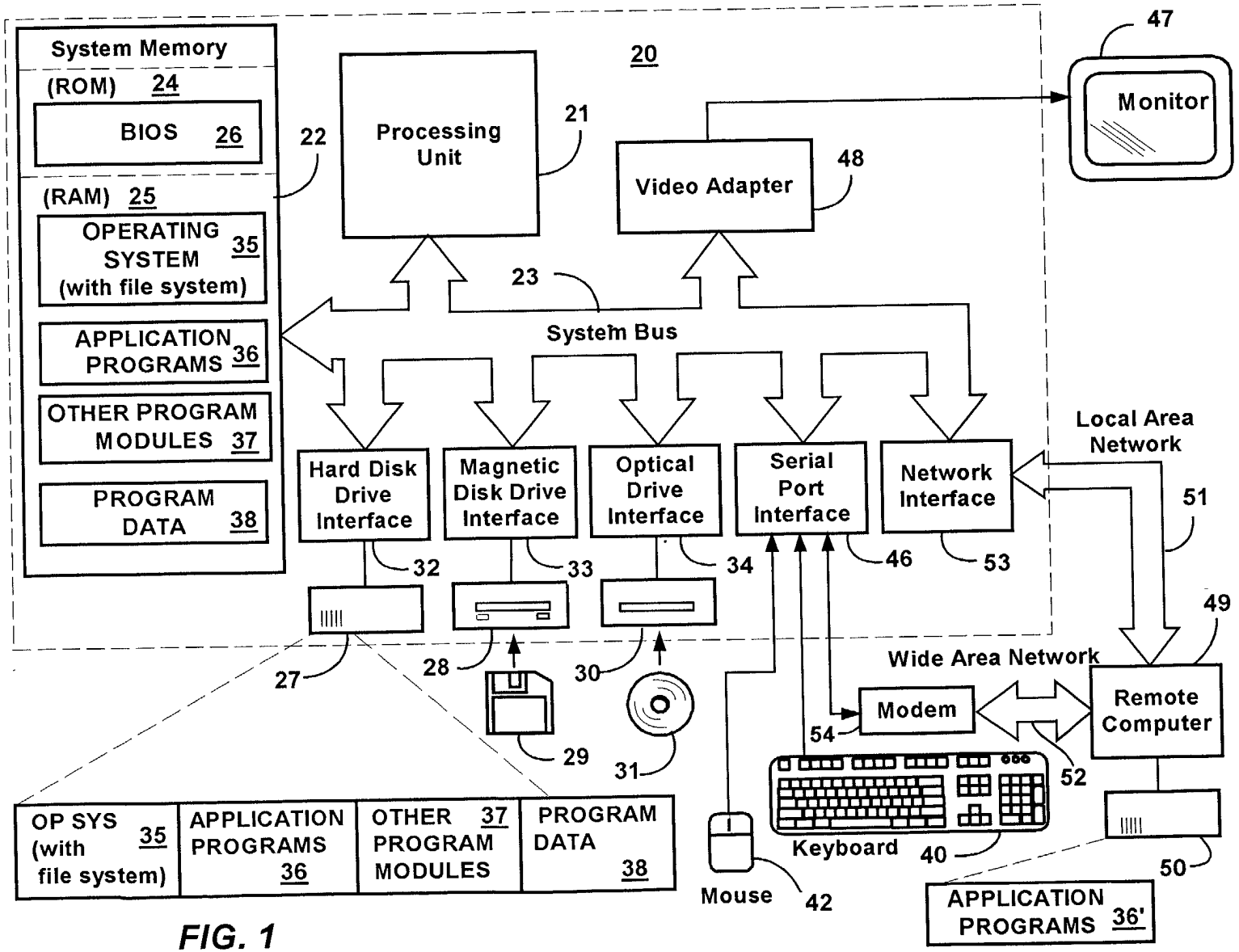

System Memory
(ROM) 24
BIOS 26
(RAM) 25
OPERATING SYSTEM 35 (with file system)
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38
22
20
Processing Unit
21
Video Adapter
48
47
Monitor
23
System Bus
Hard Disk Drive Interface
Magnetic Disk Drive Interface
Optical Drive Interface
Serial Port Interface
Network Interface
32
33
34
46
53
Local Area Network
51
49
27
28
30
29
31
Wide Area Network
Modem
54
Remote Computer
52
Keyboard
40
50
Mouse
42
APPLICATION PROGRAMS 36'
OP SYS (with file system) 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38

FIG. 1

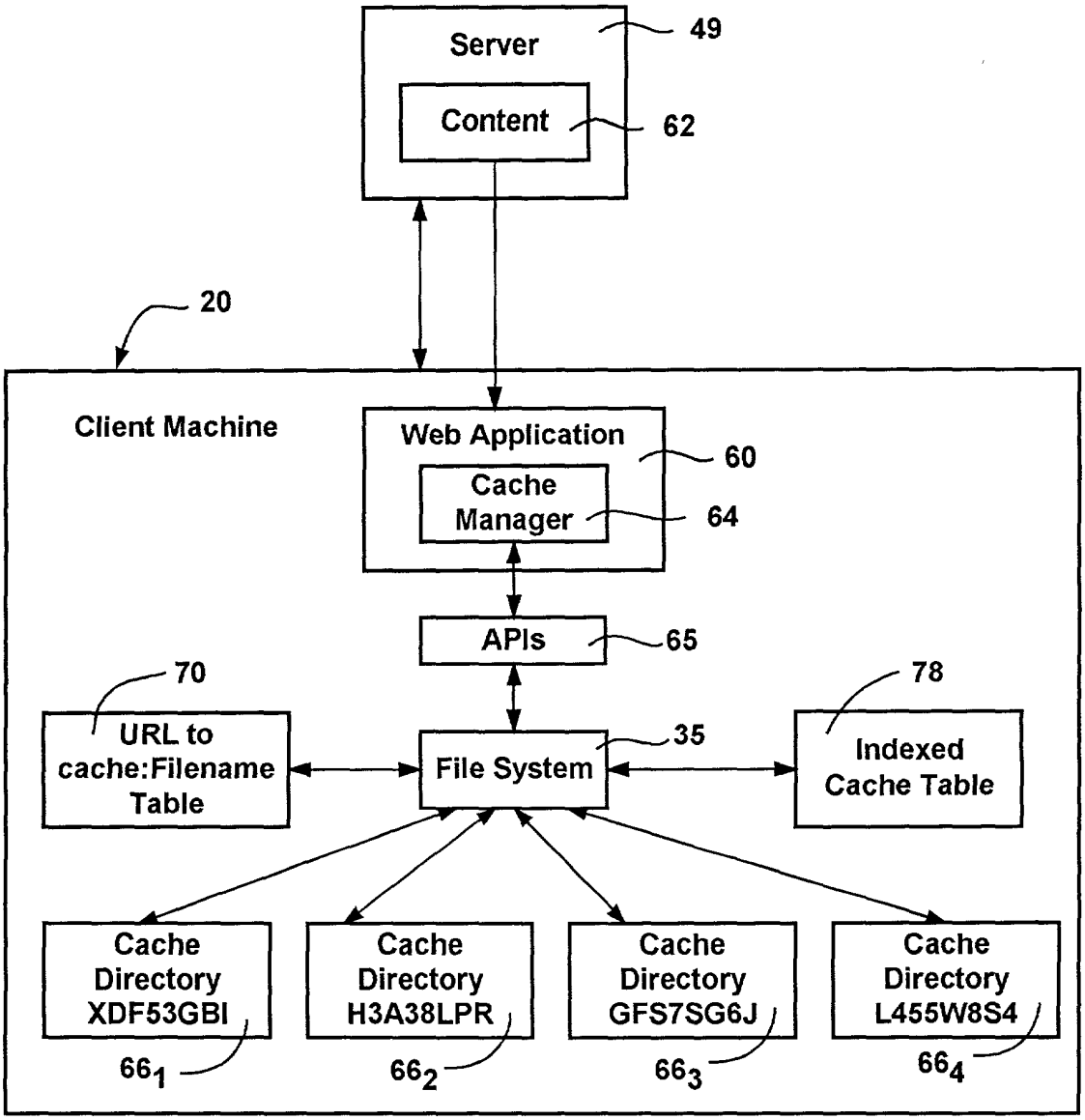

Server
49
Content
62
20
Client Machine
Web Application
60
Cache Manager
64
APIs
65
70
URL to cache:Filename Table
File System
35
78
Indexed Cache Table
Cache Directory XDF53GBI
Cache Directory H3A38LPR
Cache Directory GFS7SG6J
Cache Directory L455W8S4
66 1
66 2
66 3
66 4

FIG. 2

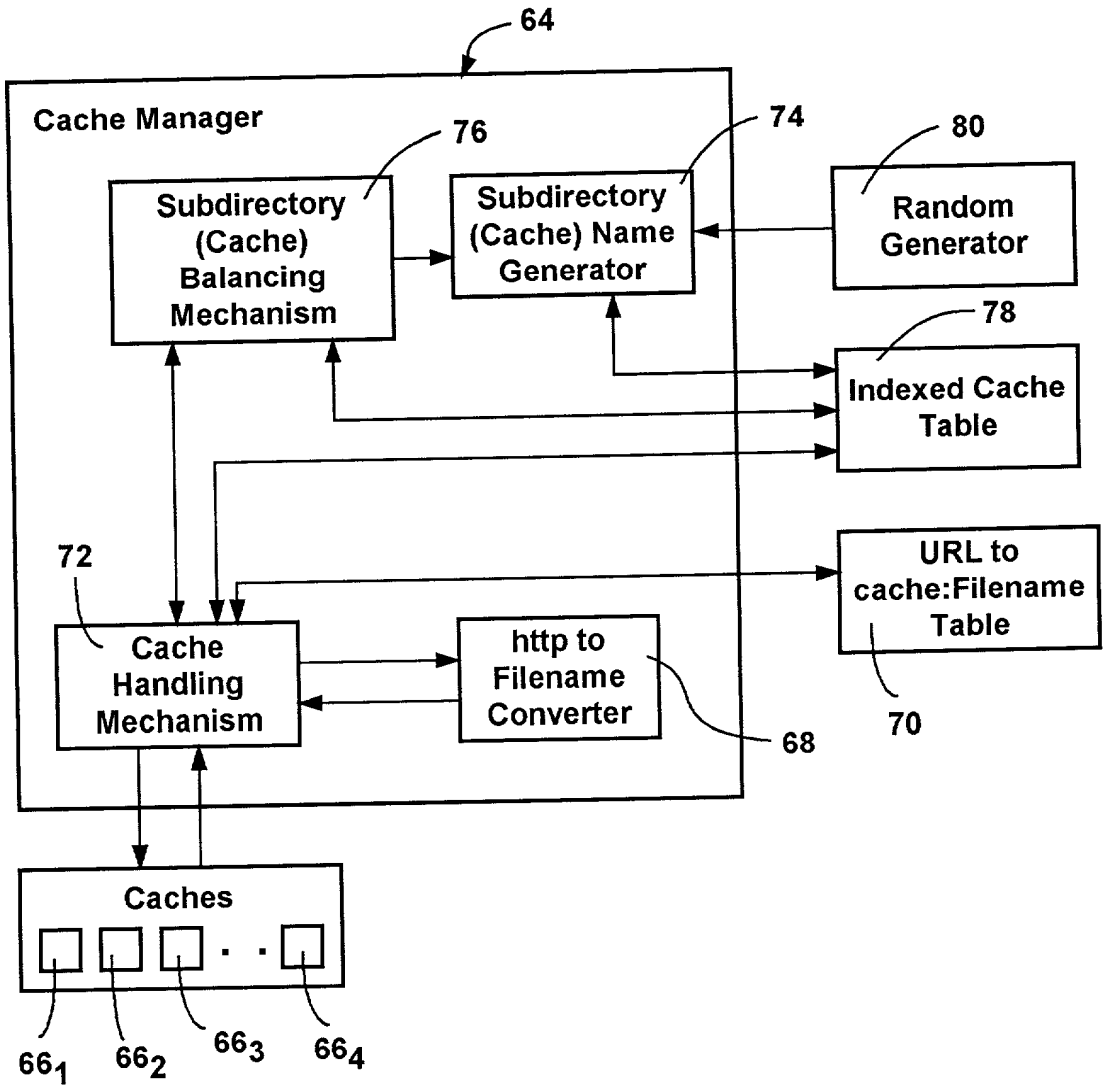

64
Cache Manager
76
74
80
Subdirectory (Cache) Balancing Mechanism
Subdirectory (Cache) Name Generator
Random Generator
78
Indexed Cache Table
72
URL to cache:Filename Table
Cache Handling Mechanism
http to Filename Converter
68
70
Caches
66 1
66 2
66 3
66 4

FIG. 3

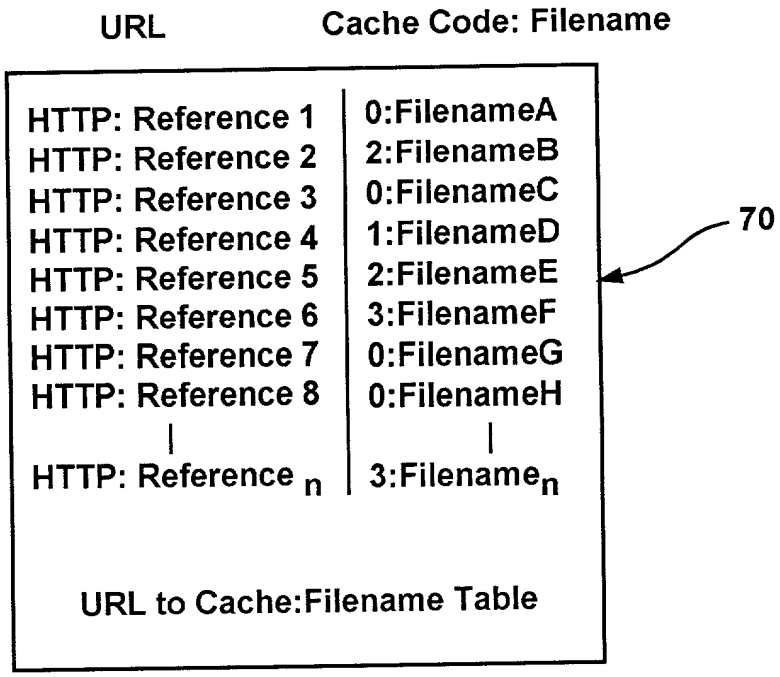

URL
Cache Code: Filename
HTTP: Reference 1
HTTP: Reference 2
HTTP: Reference 3
HTTP: Reference 4
HTTP: Reference 5
HTTP: Reference 6
HTTP: Reference 7
HTTP: Reference 8
HTTP: Reference n
0:FilenameA
2:FilenameB
0:FilenameC
1:FilenameD
2:FilenameE
3:FilenameF
0:FilenameG
0:FilenameH
3:Filename n
70
URL to Cache:Filename Table

FIG. 4

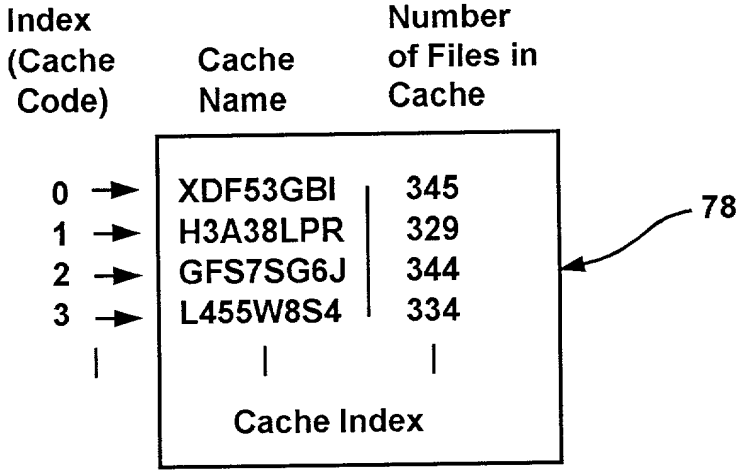

Index (Cache Code)
Cache Name
Number of Files in Cache
0
1
2
3
XDF53GBI
H3A38LPR
GFS7SG6J
L455W8S4
345
329
344
334
78
Cache Index

FIG. 6

FIG. 5

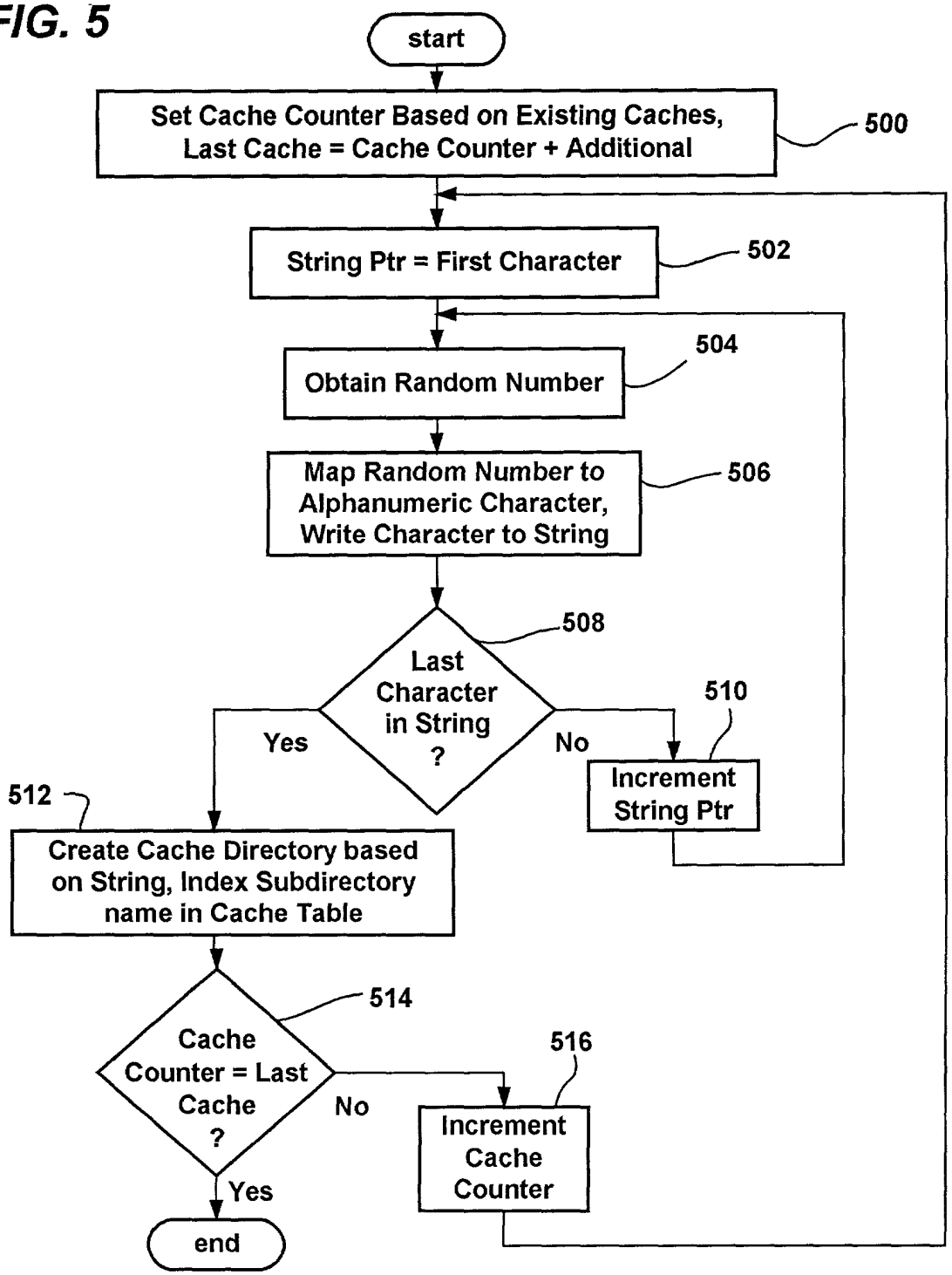
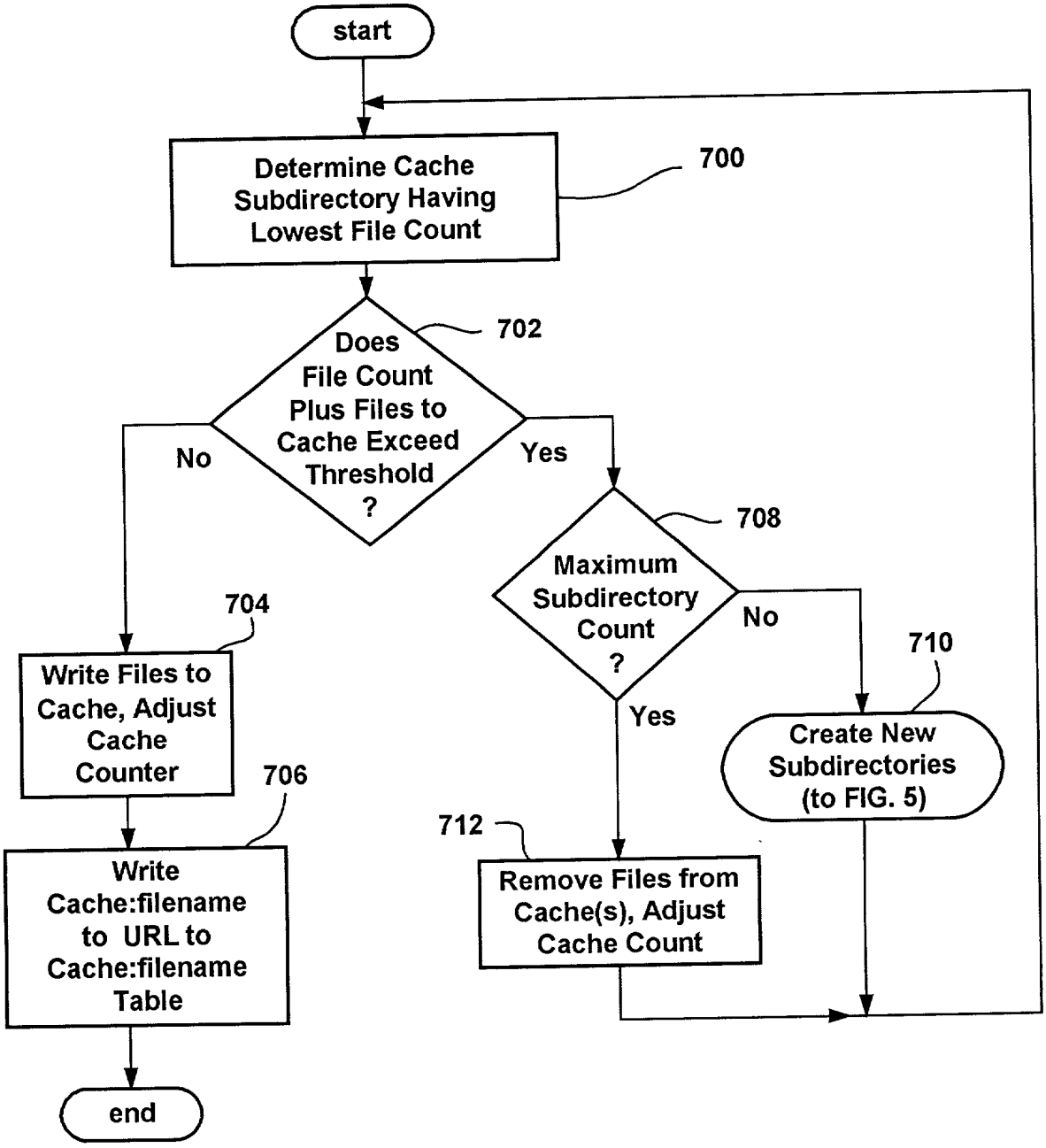

start
Set Cache Counter Based on Existing Caches, Last Cache = Cache Counter + Additional
500
String Ptr = First Character
502
Obtain Random Number
504
Map Random Number to Alphanumeric Character, Write Character to String
506
Last Character in String ?
508
Yes
No
Increment String Ptr
510
Create Cache Directory based on String, Index Subdirectory name in Cache Table
512
Cache Counter = Last Cache ?
514
No
Yes
Increment Cache Counter
516
end

FIG. 7


start
Determine Cache Subdirectory Having Lowest File Count
700
Does File Count Plus Files to Cache Exceed Threshold ?
702
No
Yes
Write Files to Cache, Adjust Cache Counter
704
Write Cache:filename to URL to Cache:filename Table
706
end
Maximum Subdirectory Count ?
708
No
Yes
Create New Subdirectories (to FIG. 5)
710
Remove Files from Cache(s), Adjust Cache Count
712

COMBINED DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that

My residence, post office address, and citizenship are as stated below next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: "Dynamic Network Cache Directories," the specification of which is attached hereto unless the following box is checked:

☐ was filed on as United States Application Serial No. or PCT International Application No. _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 C.F.R. § 1.56.

I hereby claim foreign priority benefits under 35 U.S.C. § 119(a)-(d) or § 365(b) of any foreign application(s) for patent or inventor's certificate, or § 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT international application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s)			Priority Not Claimed
_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	☐
_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	☐

I hereby claim the benefit under 35 U.S.C. § 119(e) of any United States provisional application(s) listed below.

_____ (Application Number)	_____ (Filing Date)
_____ (Application Number)	_____ (Filing Date)

I hereby claim the benefit under 35 U.S.C. § 120 of any United States application(s), or § 365(c) of any PCT international application(s) designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application(s) in the manner provided by the first paragraph of 35 U.S.C. § 112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 C.F.R. § 1.56 which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application.

_____ (Application Number)	_____ (Filing Date)	_____ (Status – patented, pending, abandoned)
_____ (Application Number)	_____ (Filing Date)	_____ (Status – patented, pending, abandoned)

As a named inventor, I hereby appoint the following attorney to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Albert S. Michalik, Reg. No. 37,395

Address all telephone calls to Albert S. Michalik at telephone number (425) 836-3030.

Address all correspondence to The Law Offices of Albert S. Michalik
704 - 228th Avenue NE
Suite # 193
Redmond, WA 98053

I hereby declare that all statements made herein of my own knowledge are true, that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of **sole or first inventor:** Rajeev Dujari

Inventor's signature Rajeev Dujari

Date 10 April 1998 Country of Citizenship: USA

Residence: 65 Kirkland Avenue #305, Kirkland, Washington 98033

Post Office Address: Same

Full name of **second joint inventor,** if any:

Inventor's signature _____

Date _____ Country of Citizenship:

Residence:

Post Office Address:

☐ Additional inventors are being named on separately numbered sheets attached hereto.

1410 declaration